Counting points in medium characteristic using Kedlaya's algorithm

Pierrick Gaudry, Nicolas Gürel

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Counting points in medium characteristic using Kedlaya's algorithm

Pierrick Gaudry — Nicolas Gürel

N° 4838

3 juin 2003

THÈME 2

ISSN 0249-6399

INRIA
FUTURS

Counting points in medium characteristic using Kedlaya's algorithm

Pierrick Gaudry*, Nicolas Gürel*

Thème 2 — Génie logiciel
et calcul symbolique
Projet TANC

Rapport de recherche n° 4838 — 3 juin 2003 — 11 pages

Abstract: Recently, many new results have been found concerning algorithms for counting points on curves over finite fields of characteristic p, mostly due to the use of p-adic liftings. The complexity of these new methods is exponential in p therefore they behave well when p is small, the ideal case being $p = 2$. When applicable, these new methods are usually faster than those based on SEA algorithms, and are more easily extended to non-elliptic curves. We investigate more precisely this dependance on the characteristic, and in particular, we show that after a few modifications using fast algorithms for radix-conversion, Kedlaya's algorithm works in time almost linear in p. As a consequence, this algorithm can also be applied to medium values of p. We give an example of a cryptographic size genus 3 hyperelliptic curve over a finite field of characteristic 251.

Key-words: Cryptography, hyperelliptic curves, Kedlaya's algorithm, point counting

* Laboratoire d'Informatique (CNRS/FRE 2653), École polytechnique, 91128 Palaiseau Cedex, France et Action TANC, INRIA Futurs, {gaudry,gurel}@lix.polytechnique.fr

Unité de recherche INRIA Futurs
Parc Club Orsay Université, ZAC des Vignes,
4, rue Jacques Monod, 91893 ORSAY Cedex (France)
Téléphone : +33 1 72 92 59 00 — Télécopie : +33 1 72 92 59 ??

Extension de l'algorithme de Kedlaya en caractéristique moyenne

Résumé : Les progrès récents des algorithmes pour compter le nombre de points d'une courbe définie sur un corps fini de caractéristique p, sont essentiellement dus aux lifts p-adic. La complexité de ces nouvelles méthodes est, en général, exponentielle en p et sont donc applicables lorsque p est petit, le cas idéal étant $p = 2$. Les méthodes basées sur le lift p-adique présentent deux avantages : dans le cadre de leurs applications, elles sont plus rapides que les algorithmes basés sur SEA d'une part et elles se généralisent plus facilement à des courbes non-elliptiques d'autre part. Nous nous intéressons ici à la dépendance en p de l'algorithme de Kedlaya. Plus particulièrement, nous montrons, après quelques modifications, que cette dépendance est linéaire en la caractéristique. Ceci a pour conséquence que cette méthode s'applique lorsque p est moyen. Nous illustrons nos résultats avec des exemples et nous donnons une courbe de genre 3, cryptographiquement sûre, sur un corps de caractéristique 251.

Mots-clés : Cryptographie, courbes hyperelliptique, algorithme de Kedlaya, comptage de points

1 Introduction

Computing the zeta function of curves over finite fields is an important task for cryptography. Indeed for the design of a cryptosystem whose security is based on the discrete logarithm problem in the Jacobian of a curve, it is required that its group order is a prime (or a small cofactor times a prime).

Two classes of algorithms can be used: those based on Schoof's idea of computing the result modulo several small primes using torsion elements [Sch95, Pil90, AH01, Cou96], and those using a lift of the curve to a p-adic ring by Satoh [Sat00], Kedlaya [Ked01] and Lauder and Wan [LW]. The first family of algorithms works for any finite field whereas the second family requires the characteristic to be small. When p-adic algorithms are applicable, there are usually much faster than those based on Schoof's idea; furthermore they are more easily extended to high genus curves. For instance, Kedlaya and Lauder-Wan algorithms have complexities which are polynomial in the genus, whereas Pila's algorithm is exponential in the genus.

The p-adic algorithms can again be divided into two classes: those based on the computation of the canonical lift of the curve, and those that use an arbitrary lift. Although the subject is recent, the literature is already quite extensive. Improvements to Satoh's original algorithm for elliptic curves have been made in various places [FGH00, FGH01, Skj03, VPV01, SST, Sat02, Gau, Har]. This method was extended to genus 2 curves in characteristic 2 by Mestre [Mes00]. Variants of Kedlaya's algorithm have been designed [GG01], in particular to extend it to hyperelliptic curves in characteristic 2 [DV02, Ver02]. Also Lauder and Wan improved their algorithm for various classes of varieties [LW, Lau].

The purpose of this paper is to analyse the behaviour of the p-adic algorithms when the characteristic is not so small. We concentrate on the case of non-elliptic curves. Indeed, for elliptic curves Schoof's algorithm allows to solve the point counting problem efficiently, whereas already in genus 2 Schoof's like algorithm is much slower and more complicated [GH00, GSa]. Hence it makes sense to try to push the p-adic algorithms as far as possible in term of the characteristic.

There are also applications: some cryptosystem designers like to have curves defined over a finite field with a medium characteristic, in order for instance to have integers modulo the characteristic that fit into one or one half of a machine word [BP98].

Let us now consider the theoretical complexity of different p-adic algorithms in term of the characteristic p. This complexity is always exponential in $\log p$ but varies quite a lot. In algorithms based on the canonical lift, the p-torsion subgroup plays an important role and it seems to be impossible to avoid working modulo an ideal defining it. As a matter of fact, it has degree p^{2g} (the p-adic ring has characteristic 0), and the complexity is at least $\Omega(p^{2g})$. So even for elliptic curves, the complexity appears to be in $\Omega(p^2)$. Note also that computing the canonical lift involves an explicit representation of p^g isogenies, namely modular equations.

On the other hand the algorithm of Kedlaya has a dependance on p which is not as clear. Our main result is that after a few adaptations using fast algorithms for radix conversions, the complexity is in fact $\widetilde{O}(p)$. Recall that the notation $\widetilde{O}(N)$ means $O(N(\log N)^k)$ for some constant integer k. We then validate this complexity with some practical experiments and give a cryptographic size genus 3 curve.

2 An overview of Kedlaya's algorithm

Let $\mathcal{C}$ be a hyperelliptic curve of genus g defined by its affine equation $y^2 = \overline{f}(x)$ over a finite field $\mathbb{F}_q$ where $q = p^n$ and $\overline{f}$ is a polynomial of degree $d = 2g + 1$. In [Ked01], Kedlaya gives an explicit construction of the de Rham cohomology space H^1 of the coordinate ring of a curve associated to $\mathcal{C}$ and computes the Frobenius action on a particular basis. This computation is done in the set of overconvergent series $A^\dagger$. Kedlaya shows how to reduce those series onto the initial basis; he also estimates the precision needed. We limit ourselves to a short description of the algorithm.

Kedlaya's algorithm illustrates the principle of the Monsky-Washnitzer cohomology in the case of hyperelliptic curves. For a more general introduction to p-adic cohomology see [vdP86] or [Kob77].

2.1 Definitions and notations

The ring $\mathbb{Z}_q$. Let $\overline{P}$ be a monic polynomial which defines $\mathbb{F}_q$ as an algebraic extension of $\mathbb{F}_p$ and let P be any monic lift of degree n over $\mathbb{Z}_p$. The first step is to build the p-adic ring needed for the computation. The quotient $\mathbb{Z}_q = \mathbb{Z}_p[t]/(P(t))$ defines, up to isomorphism, the ring of integers of the unramified field extension K of degree n of $\mathbb{Q}_p$. In practice, we will be working in $\mathbb{Z}_q/p^\nu$, where ν is the p-adic precision. The output of the algorithm is the numerator of the zeta function associated to $\mathcal{C}$ modulo p^ν. Thus we can recover the numerator of the zeta function as soon as ν is larger than a quantity which depends on the Weil's bounds. One can show that $\nu = \widetilde{O}(nd)$. In the following we extend the p-th power Frobenius of $\mathbb{F}_q$ to many different objects and for simplicity we will denote it by σ regardless of the object on which it operates. In particular, σ extends to $\mathbb{Z}_q$.

The algebra $A^\dagger_+$. The lift of $\overline{f}$ to f in $\mathbb{Z}_q[x]$ defines a lift of the curve in characteristic zero. We recall that $A^\dagger$ is constructed as the quotient $\mathbb{Z}_q$-algebra of power series in x, y and $1/y$ modulo $y^2 - f$, and such that the valuation of the coefficients grows at least linearly with the degree in x, y and $1/y$. In the algorithm it suffices to consider $A^\dagger_+$, the subalgebra of $A^\dagger$ of power series in $\tau = 1/y^2$ (i.e. $A^\dagger_+$ is the subalgebra of $A^\dagger$ which is stable under the hyperelliptic involution).

For the computation, an important consequence of the fast convergence is that the precision μ in τ for which all the coefficients are zero modulo p^ν is linear in $p\nu$ (i.e. in $\widetilde{O}(pnd)$, see Lemma 2 in [Ked01] for more details). To represent an element of $A^\dagger_+$, we adopt a normalised form:

Definition 1 (Normal form) *We say that an element $S(x,\tau) = S_0(x) + \sum_{i>0} S_i(x)\tau^i \in A^\dagger_+$ is represented in normal form if $\deg(S_i) < d$ for $i > 0$.*

Notice that, using the equation $y^2 = f(x)$, we can write any element of $A^\dagger_+$ in normal form, and that $\deg(S_0)$ can be arbitrarily large. We will see below that in Kedlaya's algorithm all the elements of $A^\dagger_+$ are such that $\deg(S_0) \leq (d-1)p - (p+1)/2$.

The extension of the Frobenius action to $A^\dagger$ is chosen such that: $x^\sigma = x^p$ and $(y^\sigma)^2 = f(x)^\sigma$.

Basis of differential forms. We consider the quotient space H^1_-, of differentials of the form $S(x,\tau)dx/y$, where $S \in A^\dagger_+$, modulo the differentials which are exact. Lemma 2 in [Ked01] implies in particular that $\mathcal{B} = \{x^i dx/y, i \in [0, d-2]\}$ is a basis of H^1_- over the p-adic number field K, and that this vector space is stable under the action induced by σ.

This action of Frobenius endomorphism on differential forms is given by $(dx)^\sigma = d(x^\sigma) = px^{p-1}dx$, hence we can compute the action of σ on each element of the basis $\mathcal{B}$ of H^1_-. We have

$$(x^i dx/y)^\sigma = (x^i)^\sigma (dx)^\sigma / y^\sigma = px^{ip+p-1}dx/y^\sigma,$$

and that expression is then rewritten as a linear combination of elements of $\mathcal{B}$ by adding appropriate elements dh where $h \in A^\dagger$. Indeed those exact forms are zero in H^1_-. This is explained in details in Algorithm 1.

2.2 Kedlaya's algorithm

In the algorithm we compute the action of σ on each elements of $\mathcal{B}$, which gives the matrix of σ in this basis. The characteristic polynomial of the norm of this matrix gives us the numerator of the zeta function, modulo the p-adic precision, due to Lefschetz fixed point formulae. We describe the whole computation in Algorithm 2.

Input: $\omega = \sum_{0 \leq m \leq \mu} Q_m(x)\tau^m dx/y$, as a normalized element of $A^\dagger_+$ times dx/y.
Output: The coefficients of ω in $\mathcal{B}$.
Step i. [First reduction: write $\sum_{0 \leq m \leq \mu} Q_m(x)\tau^m dx/y$ in the form $Q(x)dx/y$]
for $k := \mu$ to 1 by -1 **do**
Compute $U_k(x)$ and $V_k(x)$ such that $Q_k(x) = U_k(x)f(x) + V_k(x)f'(x)$.
Replace the coefficient $Q_k(x)\,\tau^k \frac{dx}{y}$ in ω by

$$\left(U_k(x) + \frac{2}{2k-1}V_k'(x)\right)\tau^{k-1}\frac{dx}{y}.$$

endfor
Step ii. [Second reduction: reduce the degree of $Q(x)$]
Initialisation: $\delta \leftarrow \deg(Q)$;
while $\delta > 2g$ **do**
Compute $\widetilde{Q}$ such that $d(2x^{\delta-2g}y) = \widetilde{Q}(x)dx/y$:

$$\widetilde{Q} = \left(2(\delta-2g)x^{\delta-2g-1}f(x) + x^{\delta-2g}f'(x)\right).$$

(Note that $\widetilde{Q}$ has the same degree as Q.)
Normalize $\widetilde{Q}$ so that it has the same leading coefficient as Q.
$Q \leftarrow Q - \widetilde{Q}$
$\delta \leftarrow \deg(Q)$
endw

Algorithm 1: Cohomological reductions

Input: A curve $\mathcal{C}$ defined by $y^2 = \overline{f}$ over the finite field $\mathbb{F}_q$
Output: The numerator of the zeta function of $\mathcal{C}$
Step 0. Compute the p-adic and τ-adic precisions, the element t^σ, the lift of f and f^σ, U and V such that $Uf + Vf' = 1$;
Step 1. [Computation of $1/y^\sigma$] From equations $(y^2)^\sigma = (f(x))^\sigma$ and $y^\sigma \equiv y^p \mod p$, we see that $1/y^\sigma = \tau^{(p-1)/2}S/y$ with $S := (1 + (f(x)^\sigma - f(x)^p)\,\tau^p)^{-1/2}$. The power series S can be computed efficiently by a Newton iteration in $A^\dagger_+$;
Step 2. [Computation of the Frobenius action on $\mathcal{B}$]
for each *differential* ω_i *in* $\mathcal{B}$ **do**
Step 2.1. [Compute $\omega_i^\sigma = p\tau^{(p-1)/2}x^{ip+p-1}Sdx/y$] This is essentially the multiplication of $p\tau^{(p-1)/2}x^{ip+p-1}$ by S, as normalised elements of $A^\dagger_+$. We obtain ω_i^σ written in the form $\sum_{0 \leq k \leq \mu} Q_k\tau^k dx/y$;
Step 2.2. [Reduce ω_i^σ as a linear combination of elements of $\mathcal{B}$] We apply the Algorithm 1 to ω_i^σ;
endfch
Step 3. [Compute the characteristic polynomial $\chi(t)$ of the norm of the Frobenius] The action of the p-th power Frobenius endomorphism on the differential forms is gathered in a matrix M. The q-th power action is then obtained by computing $\mathrm{Norm}(M) = M\,M^\sigma \cdots M^{\sigma^{n-1}}$. The characteristic polynomial χ of the Frobenius is computed as the characteristic polynomial of this matrix;
return $t^{2g}\chi(1/t)$

Algorithm 2: The main algorithm

3 Complexity in p

We give a proof that the complexity of Kedlaya's algorithm is almost linear in p. We recall that we use the Soft-Oh notation, thus any contribution in $\log p$, $\log n$ or $\log d$ in the complexity is not be taken into account.

3.1 Bit-size of the different objects

Notice that a multiplication between two objects of bit-size N is assumed to take time $\widetilde{O}(N)$, thanks to Schönhage's fast multiplication algorithm. To analyse the complexity it is important to describe the bit-size of the different objects we manipulate in the algorithm.

Using the notations of Section 2.1, $\mathbb{Z}_q$ denotes the quotient $\mathbb{Z}_p[t]/(P(t))$ and elements in $\mathbb{Z}_p$ are truncated to precision p^ν where $\nu = \widetilde{O}(nd)$. This implies that an element of $\mathbb{Z}_q$ is represented as a polynomial of degree n with coefficients in $\mathbb{Z}/(p^\nu)$, therefore the bit-size of an element of $\mathbb{Z}_q$ is $\widetilde{O}(n^2 d)$. An element of the set $A^\dagger_+$, represented in normal form, is a power series in τ, truncated modulo τ^μ, over the polynomials of degree d over $\mathbb{Z}_q$. We recall that the precision in τ, namely μ, is linear in p, and that the coefficient $S_0(x)$ is of degree at most $O(dp)$. More precisely we have $\mu = \widetilde{O}(pnd)$, therefore the bit-size of an element of $A^\dagger_+$ is $\widetilde{O}(n\nu \cdot \mu d) = \widetilde{O}(pn^3 d^3)$.

ν (p-adic precision)	$\widetilde{O}(nd)$	an element of $\mathbb{Z}_q$	$\widetilde{O}(n^2 d)$
μ (τ-adic precision)	$\widetilde{O}(pnd)$	an element of $A^\dagger_+$	$\widetilde{O}(pn^3 d^3)$

Table 1: Bit-size of main elements

3.2 Frobenius substitution

To compute the Frobenius action on $\mathbb{Z}_q$ we have to estimate t^σ modulo the p-adic precision. The element t^σ is a zero of P and is congruent to $t^p \mod p$, therefore we use a Newton iteration:

$$\begin{cases} x_0 = t^p \mod p, \\ x_{i+1} = x_i - \frac{P(x_i)}{P'(x_i)}. \end{cases}$$

It costs $\log p$ operations in $\mathbb{F}_q$ for the initialisation. At the step i of the iteration we have computed $t^\sigma \mod p^{2^i}$. As usual for this type of Newton computation, the overall cost is a constant times the cost of the last step. In this case it costs $O(n)$ multiplications in $\mathbb{Z}_q$ at maximal precision. Hence, the computation of t^σ costs $O(n)$ operations in $\mathbb{Z}_q$, i.e. is in $\widetilde{O}(n^3 d)$. Let

$$z = z_{n-1}t^{n-1} + z_{n-2}t^{n-2} + \cdots + z_1 t + z_0$$

be an element of $\mathbb{Z}_q$. For $z^\sigma = \sum_{i=0}^{n-1} z_i (t^\sigma)^i$, Horner's method yields a way of computing it at a cost of n operations in $\mathbb{Z}_q$. So this computation is in time $\widetilde{O}(n^3 d)$.

More generally, computing $\sigma^k(z)$ for any k can be done at the same cost, by lifting t^{σ^k} and plugging it into the expansion of z. Consequently, using the 2-adic expansion of k, we can compute the norm of an element in time $\widetilde{O}(n^3 d)$.

3.3 Normal form

At several places in the algorithm, the elements of $A^\dagger_+$ do not come naturally in their normal form. First, this occurs in Step 1, where we have to compute the series $(f(x)^\sigma - f(x)^p)\tau^p$. Also in Step 2.1, we have to put the series $p\tau^{(p-1)/2}x^{ip+p-1}$ in normal form before multiplying it by S. More details about these steps are given below.

For small p, this normalisation step takes a negligible time and can be done in a naive way. However, for large p, using naive algorithms makes it the dominant step in the whole algorithm. It is then required to use asymptotically fast algorithms, namely the recursive radix-conversion algorithm.

More precisely, let $Q(x)\tau^m \in A_+^\dagger$ with $\deg(Q) \geq d$. Let $k = \left\lceil \frac{\deg(Q)}{d} \right\rceil$. The normal form will follow from the coefficients of the (f)-adic expansion of Q, that is the polynomials $(a_0, \ldots, a_k)$ such that

$$Q\tau^m = (a_0 + a_1 f + \cdots + a_{m-1} f^{m-1} + a_m f^m + \cdots + a_k f^k) f^{-m},$$

where all the a_i's have degree at most $d-1$. Then if we put $Q_0 = a_m + a_{m+1} f + \cdots + a_k f^{k-m}$ and $Q_i = a_{m-i}$ for $i > 0$, we get the normal form $Q(x)\tau^m = Q_0(x) + \sum_{i>0} Q_i(x)\tau^i$.

To compute the a_i's, we compute Q_1 and Q_2 such that $Q = Q_1 f^{\lfloor k/2 \rfloor} + Q_2$ and we call the radix-conversion algorithm recursively on Q_1 and Q_2. It takes $\widetilde{O}(kd)$ operations in $\mathbb{Z}_q$ to compute those coefficients (see [vzGG99, Theorem 9.15] for more details).

Conversely, building Q from its (f)-adic expansion can also be done in the same time, using a similar strategy.

3.4 Cost of cohomological reductions

First, the polynomials U and V such that $Uf + Vf' = 1$ are computed once for all. In one step of the first reduction applied to a differential $\omega_k = Q_k \tau^k dx/y$, where $\deg(Q_k) \leq d-1$, we compute $\widetilde{\omega}_k = \widetilde{Q}_{k-1}\tau^{k-1} dx/y$ in the same class of cohomology as ω_k, for $k > 0$. We refer to Algorithm 1 for the formulae that are used; they involve the multiplication of Q_k by U and V to obtain U_k and V_k, thus it costs a constant number of multiplications of polynomials of degree $O(d)$ over $\mathbb{Z}_q$, which can be performed in time $\widetilde{O}(n^2 d^2)$.

One step of the second reduction decreases by 1 the degree $\delta > d-1$ of the polynomial Q in the differential $Q dx/y$. Again, in Algorithm 1, we see that this involves one inversion and $O(d)$ multiplications of scalars (the polynomial multiplications are just shifts), thus the time complexity is $\widetilde{O}(n^2 d^2)$.

3.5 Overall complexity

In Step 1, let $U = 1 + (f^\sigma - f^p)\tau^p$. We compute the normal form to write

$$U = U_0 + U_1\tau + \cdots + U_p\tau^p,$$

where $\deg(U_i) < d-1$. It costs the computation of the polynomial $f^\sigma - f^p$ of degree $O(pd)$ and its radix conversion as described in Section 3.3. The time complexity is $\widetilde{O}(n^3 d^2 + pn^2 d^2)$.

Then we perform a Newton iteration to take the inverse of the square root of U. There are $O(\log \mu)$ iterations to be done and after each step we renormalise the series to prevent the growth of the coefficients. As usual for a Newton iteration, the overall cost is a constant times the cost of the last iteration. This last iteration involves a constant number of multiplications of two elements of $A_+^\dagger$ truncated modulo τ^μ, therefore the cost of the multiplication is $\widetilde{O}(pn^3 d^3)$ (see Table 1).

The result of this operation is a truncated power series $S = S_0 + S_1\tau + \cdots + S_\mu \tau^\mu$ with $\deg(S_i) \leq 2d$. When we compute the normal form to the coefficient in τ^k that does not actually affect the degree in x of the previous term in τ^{k-1}. This implies in particular that $S_0 = 1$. The cost of a normalisation in this case is $\widetilde{O}(pn^3 d^3)$.

The global cost of Step 1 is then $\widetilde{O}(pn^3 d^3)$.

In Step 2, we have to compute $B = p\tau^{(p-1)/2} x^{pi+p-1} S$ for $i \in [0, d-2]$. We concentrate on the worst case, namely $i = d-2$, and we write $A = p\tau^{(p-1)/2} x^{p(d-1)-1}$ and $B = AS$. After normalisation we can write

$$A = A_0 + A_1\tau + \cdots + A_k\tau^k,$$

where $k := (p-1)/2$. The degree of $A_0(x)$ in x is $k_0 = (d-2)(p+1)/2$ and the degree of S in τ is μ, are both linear in p. Naively, there are p multiplications of a polynomial of degree p by a polynomial of degree d to do. Thus the complexity for computing B appears to be quadratic in p.

A workaround to this problem, is to use the "complete" (f)-adic expansion of A and obtain

$$A = \sum_{i=-k_1}^{k} A_i \tau^i = \tau^{-k_1} \left(\sum_{i=0}^{k+k_1} A_{i-k_1} \tau^i \right),$$

with k_1 linear in p and $\deg(A_i) \leq d-1$. The time complexity of this operation as shown in Section 3.3 is $\widetilde{O}(pn^2d^2)$. Now $\tau^{k_1}A$ and S are both power series truncated at precision μ in τ with polynomials coefficient of degree $d-1$ in x. The cost of the multiplication of $\tau^{k_1}A$ by S is the same as the last step of the Newton iteration, thus still $\widetilde{O}(pn^3d^3)$. Shifting the result by τ^{-k_1}, we obtain an expression for B that involves negative powers in τ. We then convert it to the normalised form as follows: Write $B = B_1\tau^{-k_1} + B_2$, where $B_1\tau^{-k_1}$ contains the negative powers of τ. Converting $B_1\tau^{-k_1}$ back into a polynomial in x is the reciprocal of the transformation explained in Section 3.3, and can be done in time $\widetilde{O}(pn^2d^2)$. Therefore we obtain B written as $B = \widetilde{B}_1(x) + B_2$, where $\widetilde{B}_1(x)$ is a polynomial in x of degree $O(dp)$ and B_2 is a power series with polynomials of degree at most $d-1$.

cohomological reductions, we apply μ times the first reduction to reduce B_2 at a cost of $\widetilde{O}(pn^3d^3)$, the contribution of B_2 is added to $\widetilde{B}_1$. The second reduction is applied $O(dp)$ times to $\widetilde{B}_1$ to get a polynomial of degree less than $d-1$. The cost is then $\widetilde{O}(pn^2d^3)$.

The cost of treating one differential form is in $\widetilde{O}(pn^3d^3)$. There are $O(d)$ of them, therefore the overall cost of Step 2 is in $\widetilde{O}(pn^3d^4)$.

In Step 3, the dominant part is the computation of the norm of a $(d-1)\times(d-1)$ matrix in K. As described in Section 3.2 the cost of the computation of M^σ is $\widetilde{O}(n^3d^3)$, and the cost of a matrix multiplication is $\widetilde{O}(n^2d^4)$. Therefore the norm computation costs $\widetilde{O}(n^2d^4+n^3d^3)$. Classical fast algorithms for computing the characteristic polynomial of matrices lead to a complexity of $\widetilde{O}(n^2d^4)$. Hence the cost of Step 3 is $\widetilde{O}(n^2d^4+n^3d^3)$.

Putting everything together, we obtain the main theorem of this section.

Theorem 2 *The overall time complexity of the Kedlaya's algorithm is $\widetilde{O}(pn^3d^4)$. The space complexity of the algorithm is $\widetilde{O}(pn^3d^3)$.*

4 Practical experiments

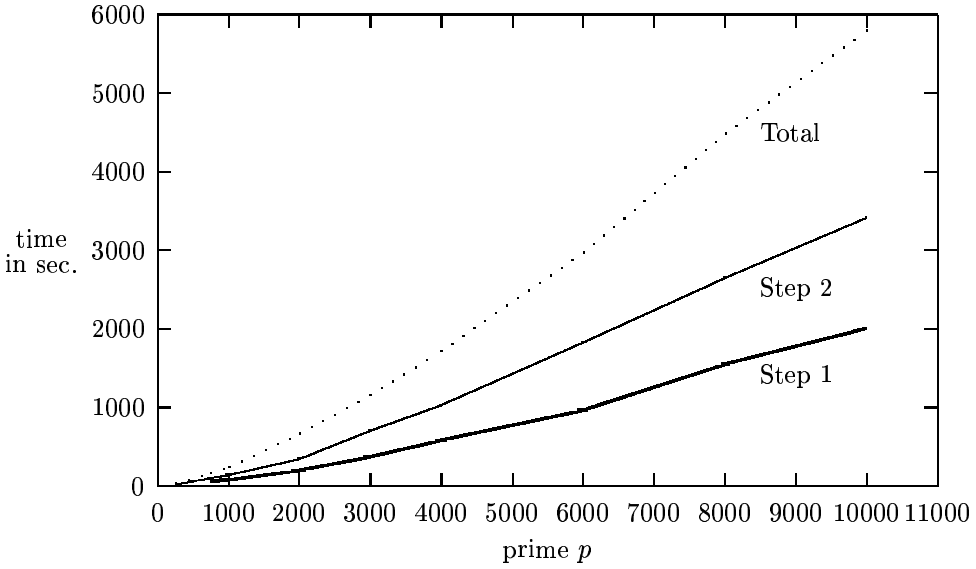
To illustrate our analysis of the complexity, we have written an implementation of Kedlaya's algorithm adapted to the medium characteristic case. In particular, we have used the recursive radix-conversion algorithm of Section 3.3. We used the NTL library written by Shoup [Sho], compiled with the GMP multiprecision package [Gra02]. Elements of the p-adic ring $\mathbb{Z}_q$ are represented as elements of an extension of the finite ring $\mathbb{Z}/p^\nu\mathbb{Z}$. The few divisions by p that occur in the algorithm are handled "by hand". NTL provides the arithmetic of polynomials over this extension ring. Above this, we added a simple arithmetic layer for the power series required in Kedlaya's algorithm. For multiplication of those power series, however, we convert everything to integers (Kronecker substitution) and use GMP's implementation of Schönhage's algorithm.

4.1 Running times when only p varies

For each prime p, we fix a root t of the polynomial used to define $\mathbb{F}_{p^3}$ over $\mathbb{F}_p$, and we consider the genus 3 hyperelliptic curves defined over $\mathbb{F}_{p^3}$ by

$$C_t \ : \ y^2 = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + t,$$

We ran our implementation on these curves for different values of p. The required p-adic precision is 7, and the power series are truncated modulo τ^μ, where μ is about 6p. The running times are given on a AMD-Athlon MP 2200+. We also measured the maximal amount of memory used by the program.



p	Minpol(t)	Step 1	Step 2	Total time	Memory
251	$t^3+20t^2+95t+116$	11 s	28 s	42 s	25 MB
503	$t^3+20t^2+95t+372$	25 s	67 s	100 s	48 MB
751	$t^3+20t^2+95t+445$	56 s	107 s	164 s	84 MB
1009	$t^3+79t^2+764t+463$	82 s	147 s	244 s	114 MB
2003	$t^3+746t^2+66t+1844$	200 s	347 s	664 s	225 MB
3001	$t^3+152t^2+1723t+2076$	371 s	712 s	1155 s	377 MB
4001	$t^3+1723t^2+2493t+3307$	585 s	1031 s	1720 s	503 MB
6007	$t^3+152t^2+3307t+3469$	971 s	1832 s	2975 s	750 MB
8009	$t^3+3469t^2+6172t+4424$	1551 s	2656 s	4482 s	1.0 GB
10007	$t^3+152t^2+3307t+3469$	2010 s	3423 s	5798 s	1.4 GB

The complexity is clearly sub-quadratic in p. We also insist on the fact that FFT-based algorithms are only efficient for very large data, and their essentially linear runtime is often hidden by logarithmic factors. In our analysis we did not take into account those logarithmic factors. In fact one factor $\log p$ is hidden in the size of the elements of $\mathbb{Z}_p$, another one is due to the cost of radix-conversion which is $O(\log p)$ multiplications, and a third $O(\log p)$ contribution comes from the complexity of Schönage's integer multiplication algorithm. Putting all together we see that there is actually a factor $O(\log^3 p)$ hidden in the complexity. This factor explains the growth of the runtime that we got.

4.2 A cryptographic size curve over $\mathbb{F}_{2^{517}}$

Let C be the hyperelliptic curve of genus 3 defined over $\mathbb{F}_{251^7}$ by the equation $y^2=f(x)$ where:

$$f(x)=x^7+t^{17808079175804175}x^6+t^{54575364231919474}x^5+t^{237357237234904}x^4+ t^{3218736229782234}x^3+t^{41232191549139817}x^2+t^{41258843266959358}x+t^{43871791627662980},$$

and t with minimal polynomial $t^7+197t^5+245t^4+t^3+148t^2+119t+225$ over $\mathbb{F}_{251}$. The characteristic polynomial of the Frobenius endomorphism is then

$$\chi(T)=T^6-s_1T^5+s_2T^4-s_3T^3+qs_2T^2-q^2s_1T+q^3$$

where

$$\begin{aligned} s_1 &= -77096895, \\ s_2 &= -482223667309721, \\ s_3 &= -1329575558557709124879171 7, \end{aligned}$$

which gives a prime cardinality of

$$N = 24725674724261831060555809558534131082659578824 2067.$$

Notice that, in this case, it takes less than 7 minutes of computation for one hyperelliptic curve and uses 190 MB of memory.

5 Concluding remarks

It is immediate to check that our analysis is still valid for the adaptation of Kedlaya's algorithm to superelliptic curves described in [GG01].

Instead of using the radix-conversion to write the polynomials as (f)-adic expansions, we could have used this basis for the whole computation. However, this would imply the finding of another appropriate basis for the differentials and rewriting the formula for the reductions accordingly. Furthermore, in practice, operations with polynomials represented in the classical basis are much faster and easier to handle.

It is known [GSb] that computing the Cartier-Manin matrix, hence also the zeta function modulo p of a hyperelliptic curve can be done at a cost $\widetilde{O}(\sqrt{p})$ (not taking into account the dependance of the other parameters). The question is still open whether this complexity can be obtained on the whole zeta function after an adaptation of Kedlaya's algorithm. Another open question is also to reduce the space complexity. Indeed, this is now clearly the bottleneck of this method.

Acknowledgements

We are grateful to Andreas Enge , Guillaume Hanrot and to François Morain for their close reading and their helpful comments on the manuscript.

The computations were carried out on the machines of the UMS Medicis at École Polytechnique.

References

[AH01] L. Adleman and M.-D. Huang. Counting points on curves and abelian varieties over finite fields. *J. Symbolic Comput.*, 32:171–189, 2001.

[BP98] D. Bailey and C. Paar. Optimal extension fields for fast arithmetic in public-key algorithms. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Comput. Sci.*, pages 472–485. Springer-Verlag, 1998.

[Cou96] J.-M. Couveignes. Computing l-isogenies using the p-torsion. In H. Cohen, editor, *Algorithmic Number Theory*, volume 1122 of *Lecture Notes in Comput. Sci.*, pages 59–65. Springer Verlag, 1996. Second International Symposium, ANTS-II, Talence, France, May 1996, Proceedings.

[DV02] J. Denef and F. Vercauteren. An extension of Kedlaya's algorithm to Artin-Schreier curves in characteristic 2. In C. Fiecker and D. Kohel, editors, *ANTS-5*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 308–323, 2002.

[FGH00] M. Fouquet, P. Gaudry, and R. Harley. An extension of Satoh's algorithm and its implementation. *J. Ramanujan Math. Soc.*, 15:281–318, 2000.

[FGH01] M. Fouquet, P. Gaudry, and R. Harley. Finding secure curves with the Satoh-FGH algorithm and an early-abort strategy. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Comput. Sci.*, pages 14–29. Springer-Verlag, 2001.

[Gau] P. Gaudry. A comparison and a combination of SST and AGM algorithms for counting points of elliptic curves in characteristic 2. Preprint, 2002.

[GG01] P. Gaudry and N. Gürel. An extension of Kedlaya's point counting algorithm to superelliptic curves. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 480–494, Berlin, 2001. Springer-Verlag.

[GH00] P. Gaudry and R. Harley. Counting points on hyperelliptic curves over finite fields. In W. Bosma, editor, *ANTS-IV*, volume 1838 of *Lecture Notes in Comput. Sci.*, pages 313–332. Springer-Verlag, 2000.

[Gra02] T. Granlund. *The GNU Multiple Precision arithmetic library – 4.1.* Swox AB, 2002. Distributed at `http://swox.com/gmp/`.

[GSa] P. Gaudry and É. Schost. Cardinality of a genus 2 hyperelliptic curve over $GF(5 * 10^{24} + 41)$. e-mail to the NMBRTHRY list, September 2002.

[GSb] P. Gaudry and É. Schost. Linear recurrence with polynomial coefficients and computation of the Cartier-Manin operator on hyperelliptic curves. In preparation.

[Har] R. Harley. Elliptic curve point counting: 32003 bits. e-mail to the NMBRTHRY list, August 2002.

[Ked01] K. Kedlaya. Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. *J. Ramanujan Math. Soc.*, 16:323–338, 2001.

[Kob77] N. Koblitz. *p-adic numbers, p-adic analysis and Zeta-functions*, volume 58 of *GTM*. Springer-Verlag, 1977.

[Lau] A. Lauder. Computing zeta functions of Kummer curves via multiplicative characters. Preprint 2002.

[LW] A. Lauder and D. Wan. Counting points on varieties over finite fields of small characteristic. Preprint 2001.

[Mes00] J.-F. Mestre. Utilisation de l'AGM pour le calcul de $E(F_{2^n})$, Décembre 2000. Lettre adressée à Gaudry et Harley, available in French at `http://www.math.jussieu.fr/~mestre/`.

[Pil90] J. Pila. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Math. Comp.*, 55(192):745–763, October 1990.

[Sat00] T. Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *J. Ramanujan Math. Soc.*, 15:247–270, 2000.

[Sat02] T. Satoh. On p-adic point counting algorithms for elliptic curves over finite fields. In C. Fieker and D. R. Kohel, editors, *ANTS-V*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 43–66. Springer–Verlag, 2002.

[Sch95] R. Schoof. Counting points on elliptic curves over finite fields. *J. Théor. Nombres Bordeaux*, 7:219–254, 1995.

[Sho] V. Shoup. *NTL: A library for doing number theory.* Distributed at `http://www.shoup.net/ntl/`

[Skj03] Berit Skjernaa. Satoh's algorithm in characteristic 2. *Math. Comp.*, 72:477–487, 2003.

[SST] T. Satoh, B. Skjernaa, and Y. Taguchi. Fast computation of canonical lifts of elliptic curves and its application to point counting. Preprint 2001.

[vdP86] M. van der Put. The cohomology of Monsky and Washnitzer. *Mém. Soc. Math. France*, 23:33–60, 1986.

[Ver02] F. Vercauteren. Computing Zeta functions of hyperelliptic curves over finite fields of characteristic 2. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Comput. Sci.*, pages 369–384. Springer-Verlag, 2002.

[VPV01] F. Vercauteren, B. Preneel, and J. Vandewalle. A memory efficient version of Satoh's algorithm. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Comput. Sci.*, pages 1–13. Springer-Verlag, 2001.

[vzGG99] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, 1999.

Unité de recherche INRIA Futurs
Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399